VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2010 **Petr Závodný**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Absolvování individuální odborné praxe
Individual Professional Practice in the Company

2010 **Petr Závodný**

Prohlášení studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě: …….. Podpis: ……..

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.“

……..……..

Abstrakt

Tato práce popisuje průběh odborné bakalářské praxe, kterou jsem absolvoval ve firmě Elcom a.s. V úvodu pojednávám o tom, čím se firma zabývá, jaké bylo mé pracovní zařazení, na jakých úkolech jsem pracoval. Dále popisuji způsob řešení úkolů a nástroje, které jsem při plnění svých úkolů používal. V závěru práce uvádím znalosti, které jsem využil a naopak znalosti, které mi při praxi chyběly a musel jsem si je doplnit.

Klíčová slova

Atribut, Blokový diagram, CMW, CVI, Čelní panel, Elcom a.s., ETL, LabView, LabWindows, TestStand, Testování ovladače, Odborná praxe, Ovladač, Přístrojový ovladač, VI, Virtuální instrumentace, VXI Plug & Play, Vývoj přístrojových ovladačů, ZVL

Abstract

This thesis describes the experience of my individual professional practice, which I passed in Elcom corp. In the introduction, I discuss in what the company is involved, what was my grade and tasks on which I worked. Next part describes the way and tools, which I was using to resolve my tasks. In the end of my thesis, I mention the knowledge which I used and which was missed during my practice.

Keywords

Attribute, Block diagram, CMW, CVI, Developing Drivers, Elcom .a.s, ETL, Front panel, Instrument Driver, LabView, LabWindows, Professional practice, TestStand, Testing Drivers, VI, Virtual Instrumentation, VXI Plug & Play, ZVL

Seznam použitých symbolů a zkratek

CMW - Wideband Radio Communication Tester

CVI - C for Virtual Instrumentation

ETL - TV Analyzer

GPIB - General Purpose Interface Bus

IVI - Interchangeable Virtual Instrument

MIKT - Měření v informačních a komunikačních technologiích

NI - National Instruments

PCI - Peripheral Component Interconnect

PXI - PCI eXtensions for Instrumentation

TCP/IP - Transmission Control Protocol / Internet Protocol

USB - Universal Serial Bus

VI - Virtual instrument

VISA - Virtual Instrument Software Architecture

VXI - VMEbus eXtensions for Instrumentation

ZVL - Vector Network Analyzer

Obsah

1 Úvod

Svou odbornou praxi jsem absolvoval ve firmě Elcom a.s. (dále jen Elcom), kde jsem spolupracoval při vývoji přístrojových ovladačů.

Firma Elcom podniká v oblastech silnoproudé elektrotechniky, elektroenergetiky a virtuální instrumentace. Podle oboru činnosti je rozdělena do pěti divizí (1):

- *Divize Aplikovaná elektronika*- Zaměřena na výzkum, vývoj a výrobu speciálních výkonových elektronických zařízení, zejména speciálních napájecích zdrojů.
- *Divize Pohony*- Zaměřena na dodávky elektromotorů, hlavně v nevýbušném provedení, např. pro petrochemický průmysl.
- *Divize Realizace a inženýrink*- Věnuje se konkrétním dodávkám rozvoden, kompenzačních zařízení nízkého a vysokého napětí.
- *Divize Výroba*- Slouží jako výrobní závod pro ostatní divize a dále pak jako materiálně-logistická centrála firmy.
- *Divize Virtuální instrumentace*- Zabývá se systémovou integrací, návrhem a dodávkami měřicích a testovacích pracovišť postavených na bázi virtuální instrumentace.

Mé pracovní zařazení bylo v divizi Virtuální instrumentace, v oddělení vývoje softwarových přístrojových ovladačů. Divize Virtuální instrumentace sídlí ve Vědecko-technologickém parku v Ostravě Porubě a spolupracuje s Vysokou školou báňskou. Tato divize je významným systémovým integrátorem v oblasti měřicích a testovacích systémů založených na propojení výpočetní a měřicí techniky. (2) Právě díky úzkému spojení s měřící a výpočetní technikou zde také probíhá vývoj přístrojových ovladačů podle IVI a Plug & Play specifikace. Ovladače jsou vytvářeny pomocí vývojových nástrojů firmy National Instruments, která nabízí ucelenou softwarovou platformu pro vývoj a testování aplikací v oblasti měřící a řídící techniky.

2 Úkoly řešené v průběhu praxe

V rámci praxe jsem se zapojil do různých fází procesu vyvíjení ovladačů pro měřicí přístroje firmy Rohde & Schwarz. Moji práci lze rozdělit na několik dílčích úkolů:

- vývoj přístrojových ovladačů
- testování vytvořených ovladačů
- vedlejší úkoly spojené s výše uvedenými činnostmi

2.1 Vývoj přístrojových ovladačů

Cílem bylo seznámit se s vývojovým prostředím LabView a s postupem vytváření přístrojových ovladačů. Pracoval jsem na následujících ovladačích:

- *ZVL (Vector Network Analyzer)*- Obvodový analyzátor
- *CMW (Wideband Radio Communication Tester)*- Širokopásový tester radiové komunikace
- *ETL (TV Analyzer)*- Analyzátor televizních signálů

Mým úkolem bylo vytvořit, podle již existujícího ovladače v LabWindows/CVI (dále jen CVI), následující části (tzv. options) ovladače ZVL pro prostředí LabView:

- Analog Demodulator (option K7)
- Bluetooth Analyzer (option K8)
- Noise Figure and Gain Measurement (option K30)

Dále provést aktualizaci LabView ovladačů ETL a CMW, tj. upravit stávající časti a přidat nové funkce podle změn v nové verzi CVI ovladače. Časová náročnost těchto úkolů byla přibližně 15 dnů.

2.2 Testování

Úkolem této fáze vývoje bylo vytvoření testovacích sekvencí pro ovladače ZVL a CMW pomocí programu TestStand. Dále pak otestovat ovladač ZVL na přístroji ZVL a opravit nalezené chyby. Časová náročnost těchto úkolů byla přibližně 30 dnů.

2.3 Vedlejší úkoly

Mezi vedlejší úkoly patřila kontrola a úprava struktury ovladače tak, aby odpovídal struktuře v CVI. Dále pak vytváření strukturovaného menu pro práci s jednotlivými funkcemi ovladače. Následně jsem pak vytvářel ukázkové aplikace v prostředí LabWindows a LabView pro ovladač CMW. Časová náročnost těchto úkolů byla přibližně 10 dnů.

3 Přístrojový ovladač

Přístrojový ovladač je množina softwarových rutin sloužících k ovládání programovatelných přístrojů. (3)Každá takováto rutina programově provede konkrétní operaci na přístroji jako například konfiguraci, čtení, zápis a spouštění přístroje. To umožňuje automatizovaně, programově pomocí počítače řídit měřicí přístroje.

Přístrojové ovladače značně zjednodušují a zefektivňují vývoj aplikací, které takto komunikují s měřicími přístroji, protože zpřístupňují jednotlivé funkce přístroje jako funkce vyšších programovacích jazyků, které jsou navíc nezávislé na použité komunikační sběrnici. Tímto je eliminována nutnost vlastními silami implementovat komunikaci s přístrojem pomocí nízkoúrovňových příkazů. Odpadá i potřeba detailních znalostí a vlastní implementace komunikačního protokolu používaného na sběrnici mezi aplikací (běžící na PC) a měřicím přístrojem.

Ovladače, na kterých jsem pracoval, jsou navrhovány a vytvářeny podle specifikace VXI Plug & Play driver. Jedná se o uznávaný a používaný průmyslový standard zastřešený VXI Plug & Play Systems Alliance. Tento standard velice detailně a striktně popisuje strukturu ovladače, názvy funkcí a jejich podobu. Obsahuje i doporučení týkající se vytváření samotného zdrojového kódu, resp. jeho vzhledu. Díky jednotné struktuře ovladačů lze snadno vytvářet aplikace pro různé přístroje od různých výrobců, případně s minimálními změnami v aplikaci měnit použité přístroje. (4) Další výhodou těchto ovladačů je otevřený kód, který umožňuje si přizpůsobit ovladač konkrétním požadavkům.

Mimo ovladačů Plug & Play se ve firmě Elcom vyvíjejí ovladače podle IVI specifikace (Interchangeable Virtual Instrument). Jedná se o vyšší třídu ovladačů, které poskytují vyšší výkon díky state caching a multithreading a umožňují zaměňovat přístroje stejného typu beze změn v původní aplikaci. (4)

3.1 LabView

Vlastní kód ovladače jsem vytvářel ve vývojovém prostředí LabView. Toto vývojové prostředí používá programovací jazyk G. Jedná se o grafický programovací jazyk. Zdrojový kód má podobu vývojového digramu, jehož objekty (jako vnitřní funkce a subVI) jsou propojeny datovými vodiči (wires). Jednotlivé aplikace nebo funkce vytvořené v LabView se nazývají virtual instrument (virtuální přístroj, zkráceně VI).

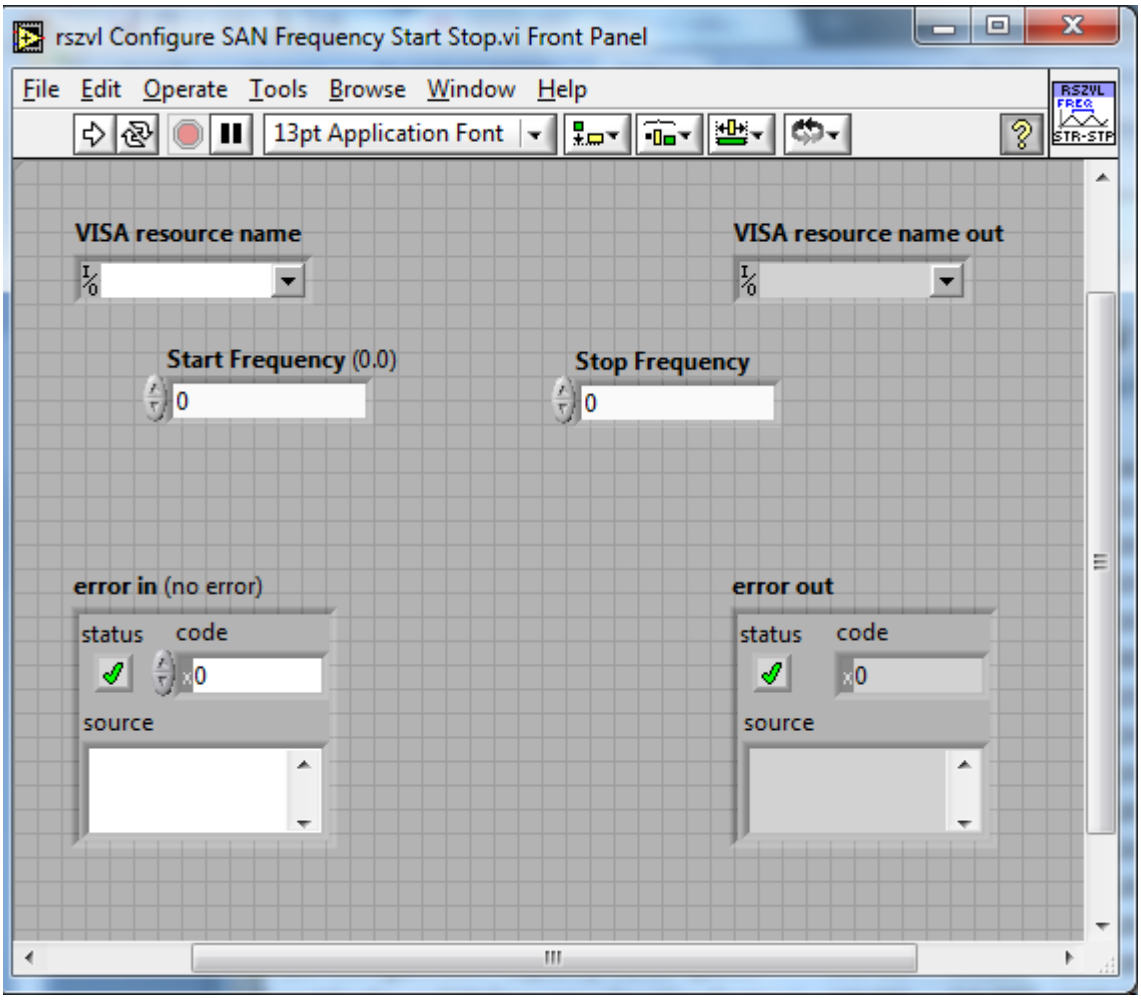
Přístrojový ovladač je tvořen množinou softwarových funkcí, takže podle terminologie LabView, je ovladač tvořen množinou virtuálních přístrojů (VI), kde každé VI reprezentuje jednu funkci přístroje neboli jeden přístroj.

Jednou z hlavních výhod LabView a ostatních aplikací od NI, např. LabWindows (obdoba LabView v jazyce C) je, že jsou zaměřeny na vývoj aplikací v oblasti měřící a řídící techniky. Pro tento účel je k dispozici množství podpůrných funkcí, z nichž pro vývoj ovladačů je velmi významná knihovna VISA (Virtual Instrument Software Architecture). VISA poskytuje programovou vrstvu mezi hardwarem a aplikací a umožňuje tak jednoduše a efektivně vytvářet aplikace nezávislé na použité platformě a sběrnici (jako je VXI, PXI, GPIB, TCP/IP, USB a sériové rozhrání).

Každá aplikace (každé VI) v LabView se skládá z čelního panelu a blokového diagramu, resp. zdrojového kódu. V následujících dvou kapitolách jsou stručně popsány tyto základní prvky a jejich součásti využívané při tvorbě ovladačů.

3.1.1 Čelní Panel

Čelní panel (Obrázek 1) tvoří vstupně/výstupní rozhraní. Jedná se o grafický ekvivalent hlavičky funkce v jazyce C. Tyto čelní panely jsou automaticky generovány na základě funkcí vytvořených CVI.



Obrázek 1: Čelní panel

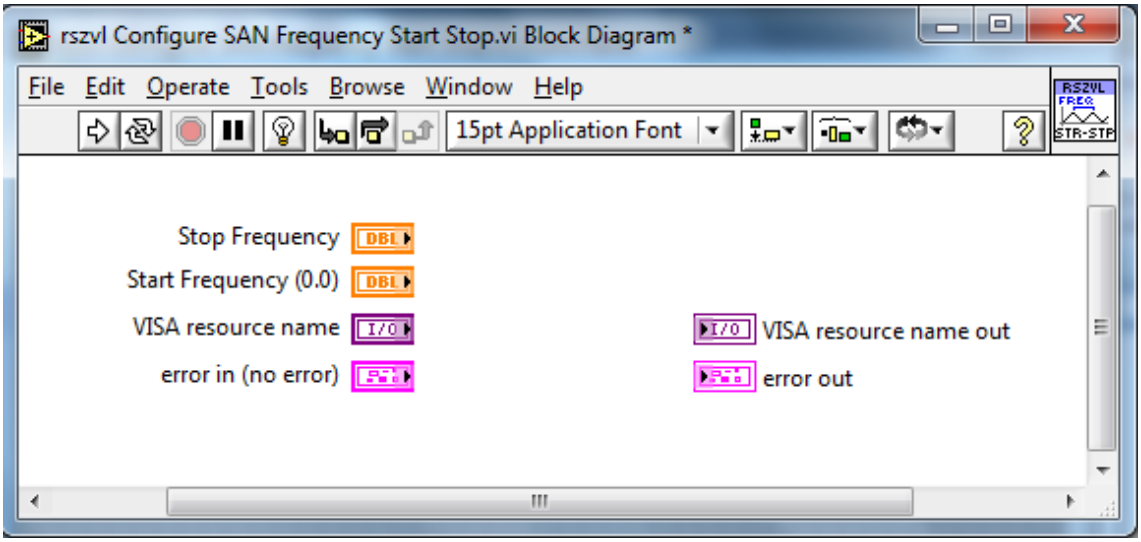
Na ukázce (Obrázek 1) je čelní panel, který slouží k nastavení frekvenčního rozsahu spektrálního analyzátoru, který je součástí ovladače ZVL. Každý čelní panel v ovladači obsahuje vždy alespoň dva páry (vstup a výstup) základních ovládacích prvků, *VISA resource name* a *Error cluster*. Tyto prvky jsou nezbytné pro správnou funkci ovladače.

VISA resource name (název zdroje VISA) obsahuje referenci na připojené zařízení. Pomocí této reference je možné přistupovat na sběrnici a komunikovat s přístrojem. Na čelním panelu je reference reprezentována jako textová hodnota udávající adresu, na které je zařízení připojeno, např. GPIB::1::1::INSTR, pro zařízení připojené přes GPIB sběrnici.

Error cluster (chybový klastr) je struktura sloužící k zachytávání chyb, která umožňuje zachytit případnou chybu, informovat o ní ostatní části ovladače a umožnit tak její další zpracování bez pádu aplikace.

3.1.2 Blokový diagram

Blokový diagram je neoddělitelnou součástí čelního panelu, obsahuje výkonný kód VI. Prvky čelního panelu a jejich hodnoty jsou přístupné přes tzv. terminály. Prázdný blokový diagram obsahuje pouze vstupní a výstupní terminály (Obrázek 2). Každý blokový digram, který je součástí ovladače, samozřejmě obsahuje základní terminály *VISA resource name* a *Error cluster*, odpovídající ovládacím prvkům čelního panelu.



Obrázek 2: Prázdný blokový diagram

4 Postup při vývoji ovladače

Vývoj ovladače probíhá podobně jako u většiny softwaru a lze jej tedy rozdělit na tři základní etapy:

- analýza a návrh
- vytváření zdrojového kódu
- testování

Posledním a finálním krokem je odeslání ovladače k certifikaci u National Instruments. Analýzou a návrhem struktury ovladače jsem se nezabýval, protože ovladače už byly navrženy a vytvořeny v CVI. Mým úkolem bylo přeprogramovat kód z CVI do LabView.

Během vývoje jsem používal nástroje od firmy National Instruments: LabView pro vývoj a TestStand pro testování.

4.1 Vytváření kódu přístrojového ovladače

Výchozím stavem pro vytváření kódu jsou prázdná VI, která byla automaticky vygenerována z CVI ovladače. Tato prázdná VI odpovídají funkcím v CVI ovladači. Mají podobný název a dokumentaci upravenou tak, aby odpovídala terminologii LabView. Adresářová struktura odpovídá stromové struktuře funkcí v CVI a ovládací prvky čelního panelu odpovídají parametrům funkcí v CVI.

Jednotlivá VI realizují svoji činnost posíláním ovládacích příkazů přes VISA do přístroje. Tyto příkazy jsou definovány výrobcem přístroje, resp. jeho firmwarem. Například pro vyčtení změřené hodnoty výkonu se pošle přístroji přes VISA příkaz:

CALC:BTO:OPOW:PEAK?

a poté se vyčte žádaná hodnota. Běžně se příkazy nesestavují přímo ve zdrojovém kódu, ale namísto toho se používají tzv. atributy.

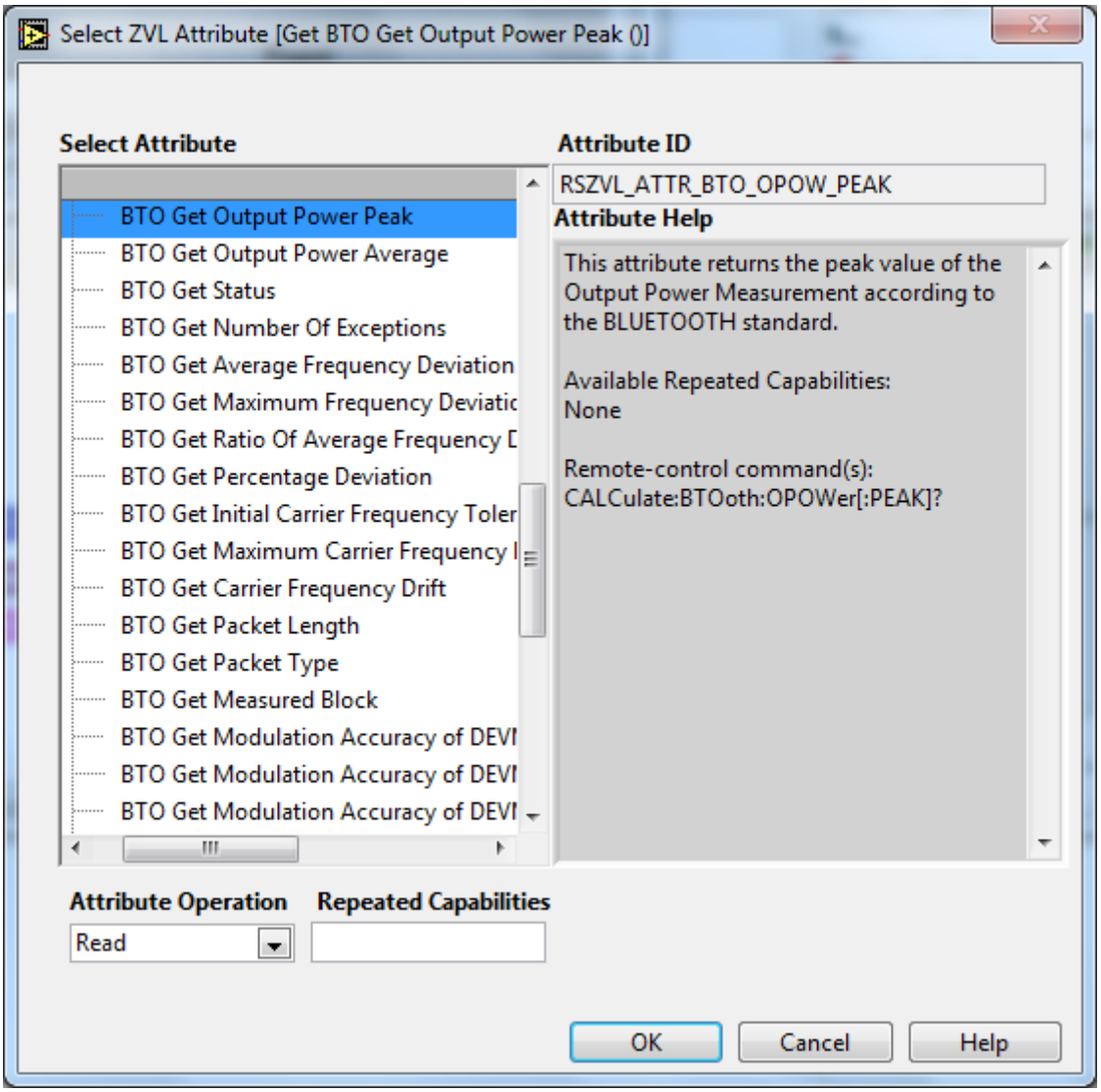
Atribut

Protože pro ovládání přístroje se používá více než tisíc ovládacích příkazů, z nichž se některé opakují, nejsou příkazy přímo vkládány do zdrojového kódu, ale používají se za tím účelem atributy.

Atributy představují elementární nastavení a funkce poskytované přístrojem, nahrazují tedy příslušný příkaz. V LabView je atribut reprezentován tzv. express VI, což je speciální druh VI používaného pro knihovní funkce. Atribut v sobě obsahuje příkaz nebo sekvenci příkazů a jejich modifikace. Například pro vyčtení změřeného výkonu lze použít tyto příkazy:

```
CALC:BTO:OPOW:AVER? MAX
CALC:BTO:OPOW:AVER? MIN
CALC:BTO:OPOW:AVER? AVER
```

Tyto tři příkazy se liší pouze v parametru, který určuje typ výsledku, a proto jsou reprezentovány pouze jedním atributem. Konkrétní příkaz je vybrán vstupním parametrem atributu. Atributy se vybírají z databáze atributů (Obrázek 3) na základě jejich názvu.



Obrázek 3: Databáze atributů

Analýza zdrojového kódu funkce v CVI

Prvním krokem bylo dohledat si ke konkrétnímu VI zdrojový kód ekvivalentní funkce vytvořené CVI. Poté jsem daný kód prohlédl a zjistil jsem, jaké atributy se volají a jaké případné další úkoly se v něm provádějí, např. kontrola vstupních dat. Dále jsem kontroloval, zda ovládací prvky čelního panelu (jejích názvy a typ) odpovídají jejich ekvivalentům v CVI a opravil případné chyby.

Na ukázce (Obrázek 4) je vidět zdrojový kód funkce pro vyčtení naměřeného výkonu Bluetooth analyzátoru (součást ovladače ZVL). Tato funkce umožňuje číst tři druhy výsledků: špičkovou hodnotu, průměrné maximu a minimum výkonu. Těmto volbám odpovídají tři *case* struktury a rovněž tři atributy, které mohou být zavolány.

Vstupním údajem této funkce je číslo udávající jaký typ výsledku požadujeme. Výstupem je reálné číslo udávající naměřený výkon.

Před vlastním voláním atributu a vyčtením hodnoty se provádí tzv. Range checking - kontrola rozsahu (podmínka před *switch*, viz Obrázek 4). Kontroluje se, zda číslo udávající typ výsledku je v očekávaném rozsahu. Pokud tomu tak není, je vybrána výchozí hodnota a tato skutečnost je ohlášena.

```
/*****************************************************************************
 * Function:    Get BTO Output Power
 * Purpose:     This function returns the output power results.
 *****************************************************************************/
ViStatus _VI_FUNC rszvl_GetBTOOutputPower (ViSession instrSession,
                                           ViInt32 type,
                                           ViReal64 *power)
{
    ViStatus    error = VI_SUCCESS;

    checkErr( Rs_LockSession (instrSession, VI_NULL));

    if ( (type<RSZVL_VAL_MEASTYPE_MIN) || (type>RSZVL_VAL_MEASTYPE_PEAK) )
        viCheckParm( RS_ERROR_INVALID_PARAMETER, 2, "Type");
    switch (type){
        case RSZVL_VAL_MEASTYPE_MIN:
            viCheckParm (rszvl_GetAttributeViReal64 (instrSession, "Min",
                         RSZVL_ATTR_BTO_OPOW_AVER, power), 3, "Power");
        break;
        case RSZVL_VAL_MEASTYPE_MAX:
            viCheckParm (rszvl_GetAttributeViReal64 (instrSession, "Max",
                         RSZVL_ATTR_BTO_OPOW_AVER, power), 3, "Power");
        break;
        case RSZVL_VAL_MEASTYPE_PEAK:
            viCheckParm (rszvl_GetAttributeViReal64 (instrSession, "",
                         RSZVL_ATTR_BTO_OPOW_PEAK, power), 3, "Power");
        break;
    }
    checkErr( rszvl_CheckStatus (instrSession));

Error:
    Rs_UnlockSession(instrSession, VI_NULL);
    return error;
}
```

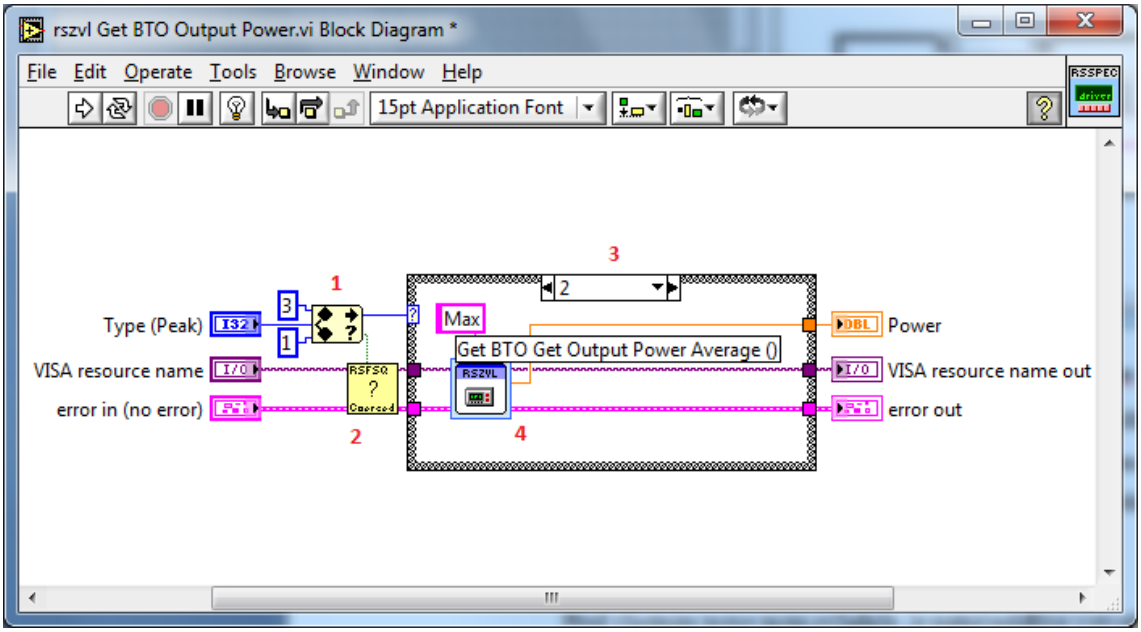
Obrázek 4: Zdrojový kód CVI

Zdrojový kód VI

I když výsledná činnosti VI musí být stejná jako je v CVI, nemusí být dosažena stejným způsobem. Většinou se ale provádí stejná posloupnost úkonů v jako v CVI. Odlišnosti jsou většinou způsobeny jednotlivými rozdíly mezi programovacími jazyky. To se nejvíce projevuje při práci s poli a strukturami, kdy CVI pracuje s ukazateli, kdežto LabView toto neumožňuje. Další rozdíly jsou způsobeny používáním rozdílných podpůrných funkcí (tzv. utility). Například pro upozornění o provedení korekce vstupu se používá v CVI funkce *viCheckParm* a LabView *VI Coercing Warning*.

Na obrázku (Obrázek 5) je zobrazen zdrojový kód VI, které plní stejný účel jako výše popsaná funkce v CVI. V tomto případě se provádí jednotlivé úkony ve stejné posloupnosti jako v CVI.

Nejdříve se kontroluje, zda je číslo udávající typ měření je v rozsahu 1 až 3 (1), pokud tomu tak není, potom se pomocí subVI (vnořené VI) correction warning (2) přidá varování do chybového clusteru. Dále se skrze *case structure* (3) vybere a zavolá příslušný atribut (4).



Obrázek 5: Zdrojový kód LabView (1) kontrola rozsahu, (2) correction warning, (3) case structure, (4) atribut

Většina VI v ovladači je podobná výše uvedenému. Liší se v počtu atributů a vstupních resp. výstupních parametrů. Složitost VI narůstá s počtem vstupních parametrů, obzvláště pokud rozsahy jednotlivých hodnot jsou závislé mezi sebou. Další odlišnou skupinu složitějších VI tvoří ta, která nepoužívají atributy, ale příkaz posílaný do zařízení se vytváří přímo ve zdrojovém kódu. Jedná se o ty VI, jejíž příkazy mají jako parametr pole (seznam) hodnot.

4.2 Testování přístrojového ovladače

Před vlastním testováním ovladače na přístroji je nutné nejdříve vytvořit sekvenci testů, které definují vlastní testování. Pro každé VI se vytvoří sekvence, která obsahuje sadu kroků a testů ověřujících funkčnost daného VI. Obecně testování probíhá tak, že se skrze testované VI nastaví v přístroji známá hodnota (popř. hodnoty) konkrétního nastavení a poté je toto nastavení opět vyčteno (pomocí atributů) a porovnáno se zadanou hodnotu. V ovladači existují

VI, která nejsou konfigurační, tj. nemění konfiguraci přístroje (např. VI spouštějící měření a manipulující s markery). U těchto VI je nemožné nebo velmi náročné kontrolovat je vůči předem známé hodnotě, proto se testují pouhým spuštěním a kontrolou, zda nedošlo k chybě za běhu VI.

4.2.1 Generování sekvencí

Prvním krokem při vytváření testovacích sekvencí je příprava souboru obsahujícího prázdné sekvence pomocí nástroje (sequence generator). Vstupem do toho nástroje je adresář obsahující všechny VI daného ovladače. Během procesu generování se pro každé VI vytvoří prázdná sekvence.

Tyto prázdné sekvence obsahují základní kroky:

- inicializaci
- spuštění daného VI
- prázdné testy vytvořené na základě ovládacích prvků čelního panelu
- a de-inicializaci.

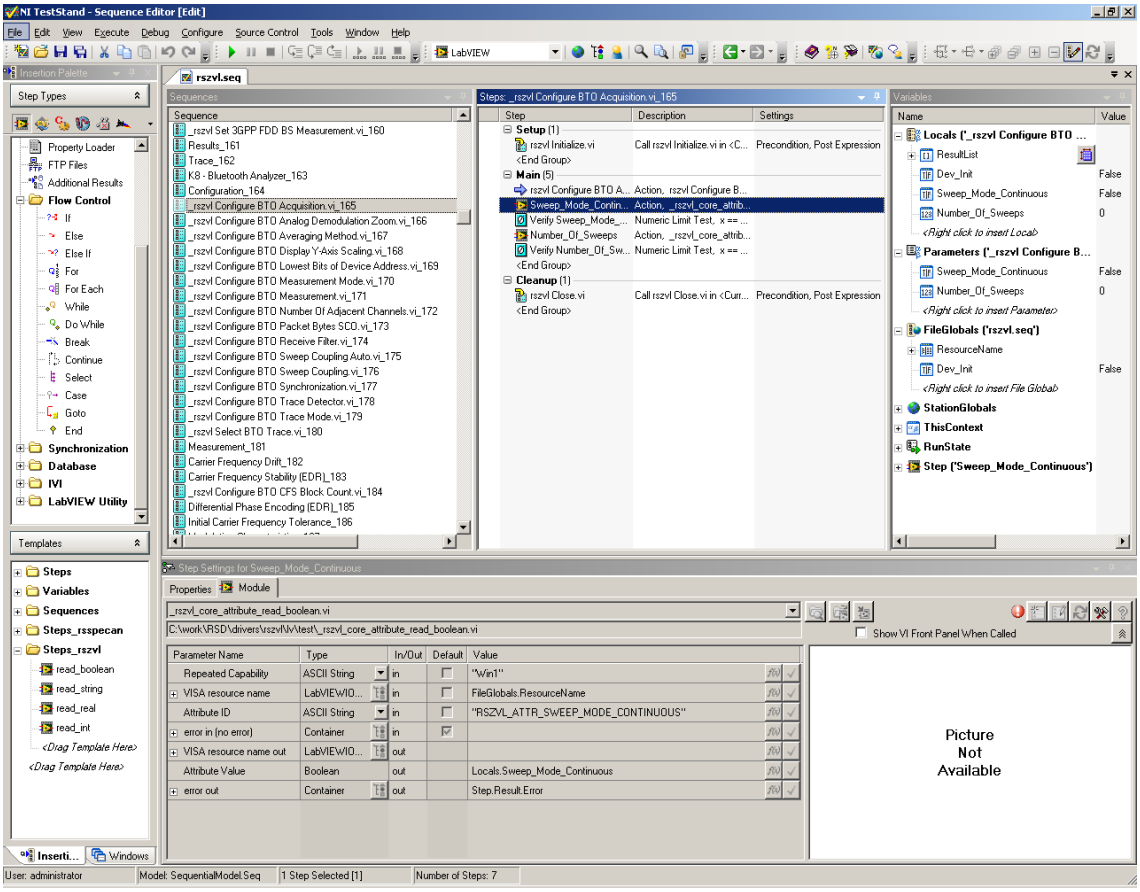
Vytváření testovacích kroků

Sekvence jsem editoval ve vývojovém prostředí TestStand (Obrázek 6). Pro každou sekvenci (Obrázek 8) jsem vytvářel její vnitřní logiku, sestavoval jednotlivé testovací kroky.

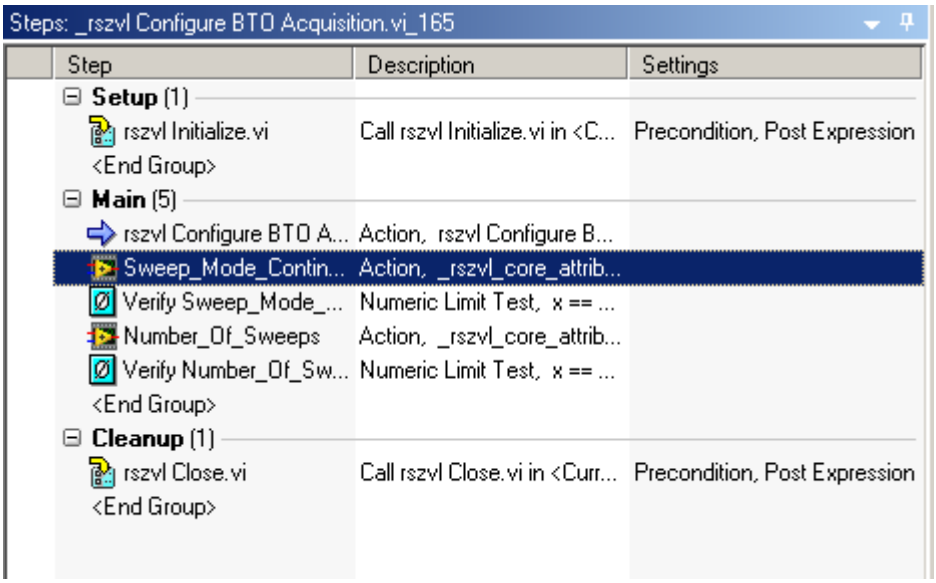
Na ukázce (Obrázek 7) lze vidět jednotlivé kroky testovací sekvence, která testuje část ovladače ZVL konfigurující akvizici. Tato sekvence se sestává z šesti kroků. První a poslední krok je společný všem sekvencím, inicializuje a ukončuje komunikaci se zařízením (sekce Setup a Cleanup).

Kroky v sekci Main jsou specifické pro konkrétní VI (sekvenci). První krok spouští testované VI se vstupními parametry, které jsem nadefinoval pro danou sekvenci. V následujícím kroku se přes pomocné VI volá atribut, kterým se opětovně vyčte nastavená hodnota parametru (atributu) *Sweep mode*. V dalším kroku se vyčtená hodnota porovnává se vstupním parametrem. Následující dva kroky provádí stejnou činnost, ale pro druhý parametr (*Number of sweeps*).

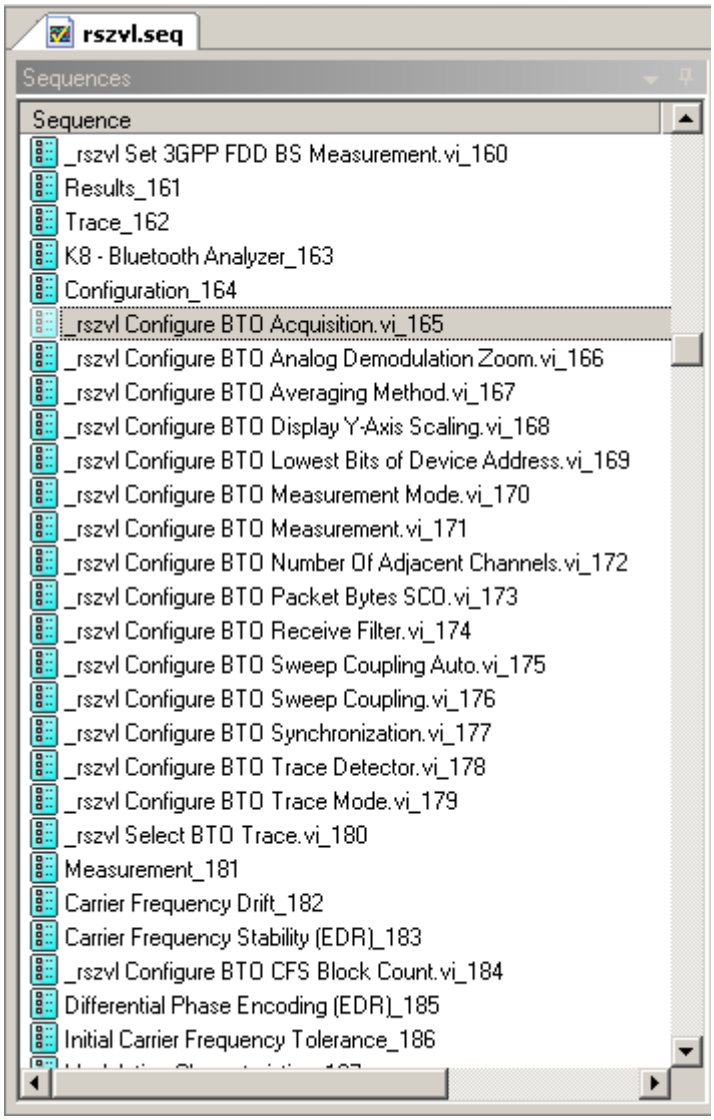
Mým úkolem tedy bylo pro každou sekvenci nadefinovat vstupní parametry testující všechny relevantní možnosti nastavení a vytvořit jednotlivé kroky v sekvenci tak, aby vyčítaly a testovaly požadované hodnoty.



Obrázek 6: TestStand



Obrázek 7: TestStand- kroky



Obrázek 8: TestStand- Sekvence

Testované parametry

Informace, na základě kterých jsem vytvářel testovací scénáře, jsem získával z dokumentace ovladače. V dokumentaci každého VI jsou uvedeny jeho vstupní parametry, rozsahy, možná nastavení a atributy, které realizují dané nastavení. Další informace jsem získával ze zdrojového kódu VI. Konkrétní hodnoty nastavení (např. frekvence, útlum apod.) jsem volil vždy jednu jako náhodnou hodnotu z rozsahu daného nastavení, druhou jako výchozí pro dané nastavení. Pokud VI obsahovalo více voleb (např. typ modulace, typ limity apod.), testoval jsem všechny jejich kombinace. Pro usnadnění těchto činností jsou k dispozici základní řídící struktury jako cykly, podmínky a skoky.

4.2.2 Testovaní na přístroji

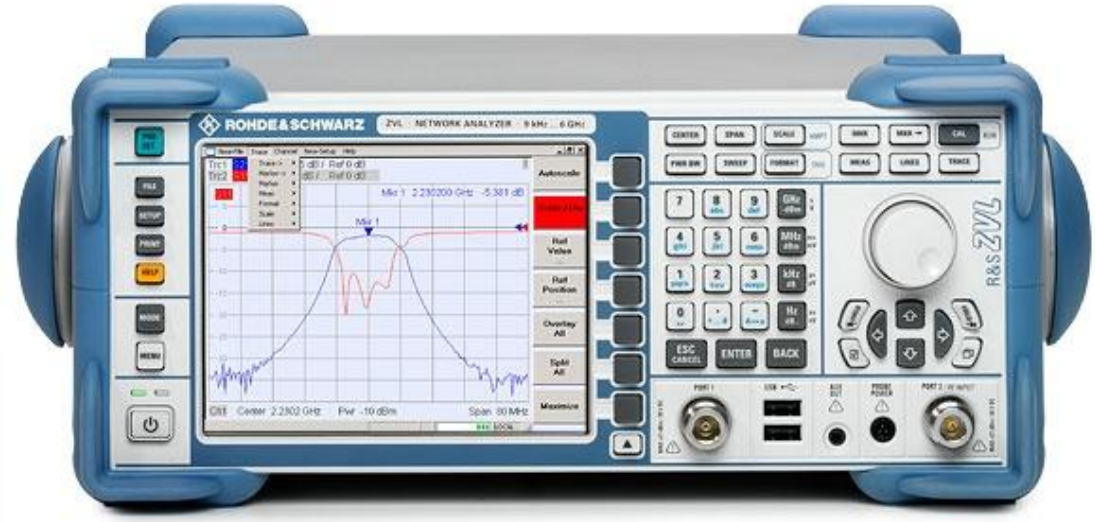
Konečnou fází vývoje ovladače je jeho otestování na přístroji, kdy se sekvence vytvořené v TestStand vyzkoušejí na skutečném přístroji (Obrázek 9). Testování probíhá na přístroji, který je připojený přes GPIB rozhraní k počítači. Připojení je realizováno prostřednictvím rozšiřující PCI karty s GPIB rozhraním. V TestStand se nastaví adresa přístroje a spustí se testovací sekvence.

Když v testovací sekvenci dojde k chybě, sekvence se v daném místě zastaví a vypíše chybové hlášení udávající typ chyby. Mým úkolem bylo najít tuto chybu, její příčinu a opravit ji. Hlavním vodítkem pro nalezení chyby je chybové hlášení, které říká, kdy k chybě došlo a o jaký typ chyby se jedná.

Například poměrně častou chybou je, že nějaká hodnota je mimo rozsah. Nalezení zdroje takovéto chyby nebývá příliš náročné, ale určení správného řešení může být už komplikovanější. Takováto chyba může mít totiž několik příčin. Může se jednat o špatně zvolenou hodnotu v testovací sekvenci nebo chybu v ovladači.

Správnost testovací sekvence jsem kontroloval vůči dokumentaci a při případných rozporech jsem konkrétní nastavení zkoušel manuálně nastavit přímo v přístroji.

Při hledání chyby ve zdrojovém kódu, jsem kontroloval kód vůči CVI a dokumentaci ovladače. Přitom jsem se zaměřoval na kontrolu správnosti atributu. Stává se totiž, že je buďto použit nesprávný atribut, případně s nesprávným parametrem, nebo je chyba uvnitř atributu.



Obrázek 9: Přístroj ZVL

4.3 Řešení vedlejších úkolů

Vedlejšími úkoly byly práce spojené s finální úpravou ovladačů. Součástí ovladače je menu, které se po instalaci ovladače integruje do LabView. Toto menu obsahuje všechny VI ovladače seskupené do stromové struktury dalších sub-menu. Strukturu menu jsem vytvářel podle stromu funkcí v CVI.

Dalším úkolem bylo vytváření ukázkových aplikací pro CMW ovladač. Ukázkové aplikace jsem vytvářel pro CVI a LabView. Byl zadán typ měření, ukázky nastavení přístroje a posloupnosti ovládacích příkazů použitých při konfiguraci. Na základě těchto informací jsem si dohledal v dokumentaci ovladače funkce, které realizují jednotlivá nastavení. Poté jsem vytvořil uživatelské rozhraní, pomocí kterého se dá manipulovat s těmito funkcemi (konfigurovat měření) a spouštět samotné měření.

5 Uplatněné znalosti

Během praxe jsem uplatnil praktické a teoretické znalosti hlavně z těchto předmětů:

- Virtuální instrumentace I a II
- Měření v informačních a komunikačních technologiích (MIKT)
- Přenos dat

Při vytváření ovladačů jsem využil praktické znalosti programování v prostředí LabView, které jsem nabyl během studia předmětu Virtuální instrumentace I. Z navazujícího předmětu Virtuální instrumentace II, jsem uplatnil znalosti a dovednosti z oblasti významu a používání přístrojových ovladačů, práce s měřícími kartami v LabView, hardware používaný při virtuální instrumentaci.

Během testování jsem využil znalosti z oblasti digitální modulace. Jednalo se hlavně o základní principy digitální modulace, základní pojmy a druhy modulací. Tyto znalosti jsem získal hlavně z předmětů MIKT a Přenos dat. Přestože při vytváření testovacích sekvencí není příliš zapotřebí těchto znalostí, protože jde hlavně o mechanické otestování funkčnosti, je dobré mít alespoň obecnou představu o testovaných funkcích, resp. jejich vstupních hodnotách. Umožnilo mi to vytvářet testy se smysluplnějšími daty a usnadnilo následné ladění ovladače.

Získané znalosti jsem nejvíce využil při testování na přístroji. I když byly pouze obecné, umožnily mi v některých případech snadněji odhalovat chyby způsobené špatnou kombinací nebo posloupností nastavovaných parametrů.

Dále jsem velmi ocenil praktické zkušenosti nabyté v předmětu MIKT při měření ukázkových úloh pomocí spektrálního analyzátoru. Během těchto měření jsem si osvojil základní návyky při manipulaci s přístroji tohoto typu. Tyto znalosti jsem uplatnil při testování ovladačů, přesněji řečeno při testování jejich částí ovládajících spektrální analyzátor.

6 Scházející znalosti

V průběhu praxe mi chyběly hlubší znalosti o způsobu práce s digitální modulací. Tyto znalosti mi nejvíce scházely pří testování na přístrojích, kdy jsem neznal přesný význam některých detailních voleb. Pokud totiž nastala při testování chyba, při které se určitá volba nenastavila korektně nebo hodnota parametru byla mimo rozsah, nemuselo se jednat o chybu ovladače, ale o nesprávné použiti volby nebo významu hodnoty této volby.

S nedostatkem znalostí o digitální modulaci souvisely i nedostatečné praktické zkušenosti při manipulaci s přístroji typu vektor signál analyzátor. Ocenil bych nějaká cvičná měření, při nichž by byla možnost osvojit si manipulaci s tímto typem přístroje.

Dále mi scházely zkušenosti z práce v prostředí TestStand, ale díky jednoduchosti ovládání tohoto programu (vycházejícího z LabView), nepovažuji toto za důležité. Chybějící znalosti jsem získal formou krátkého školení od mého konzultanta. Totéž platí i pro práci s prostředím CVI, kde se vychází z jazyka C, propojeného s virtuální instrumentací. (V jazyce c jsem programoval na střední škole.).

7 Závěr

Během své praxe jsem se seznámil s procesem vývoje přístrojových ovladačů a úspěšně jsem se zapojil do jejich vývoje. Bylo mi umožněno využít znalosti z oblasti telekomunikační techniky a programování získané dosavadním studiem. Protože praxe probíhala souběžně s výukou, zaměřil jsem se při studiu podrobněji na témata související s mou praxí, kterým bych za normálních okolností zřejmě nevěnoval takovou pozornost. Dále mi tato zkušenost pomohla doplnit nedostatky v mých znalostech a zjistit tak na jaké oblasti se zaměřit v budoucím studiu.

V průběhu testování ovladačů mi bylo umožněno seznámit se s velice sofistikovanými měřicími přístroji, se kterými jsem se v běžné výuce nesetkal a doplnit si tak praktické zkušenosti při práci s těmito přístroji.

Literatura

1. **Elcom a.s.** O společnosti. *Elcom a.s.* [Online] 2010. http://www.elcom.cz/o-spolecnosti-elcom/.

2. **Elcom a.s.** Náplň divize Virtuální instrumentace. *Elcom a.s.* [Online] 2010. http://www.elcom.cz/virtualni-instrumentace/napln-divize/.

3. **National Instruments.** What is an Instrument Driver? *NI Developer Zone.* [Online] 4. 5 2009. http://zone.ni.com/devzone/cda/tut/p/id/4803.

4. **National Instruments.** Types of Instrument Drivers. *NI Developer Zone.* [Online] 19. 5 2009. http://zone.ni.com/devzone/cda/tut/p/id/4803#toc0.

Seznam obrázků